CS 363 Lab #5 – Block variable allocation

In today's lab we will look at allocating variables to registers. As we saw in class this week, languages such as C and Java allow you to declare variables inside a block of code like a loop. These are "very" local variables that are only defined for that block. Variables declared outside the block are less local, but are still in scope.

We are interested in knowing how many memory locations or registers will be required for all our local variables. It turns out that we can save some memory if we note that variables declared in mutually exclusive blocks can share the same memory location. This knowledge can be used by a compiler to minimize the number of registers it must allocate or the amount of space to reserve on the run-time stack for an activation record.

Today, you'll complete a program to simulate this scenario. Ask the user to enter the name of an input file (e.g. input.txt) and print all output to the screen. The input file will be an abstraction of a function with blocks and variable declarations. The only things you will see in the input are curly braces and "declare" statement declarations, like this:

```
{
  declare a;
  declare b;
  declare c;
  {
    declare x;
    declare y;
  }
  {
    declare r;
  }
}
```

In this case, the variable r can share space with x because we never need both variables at the same time. In your output just list the memory locations (or registers) one per line listing all the variables that can occupy that space. Here is the intended output that corresponds to the above input example:

```
Register  1:  a(1)
Register  2:  b(1)
Register  3:  c(1)
Register  4:  x(2), r(2)
Register  5:  y(2)
```

The number in parentheses indicates the nesting level of the variable. The name of the variable as well as its nesting level are given by the toString() method in the Variable class.

I've started the program with Lab5.java and Variable.java located on the server. So far it handles input, but that's about it. The output phase of the program won't print anything because we have not yet allocated any variables to registers.

Test your program using the example input files on the server.